ICEE 2012

WS #5:

Database as a CS/E Competency:

A Software Engineering Attribute

ICEE DESCRIPTION OF WORKSHOP

Workshop facilitator: Dr. Fred Springsteel, Emeritus Professor, Missouri University

PhD, U Washington, Seattle

Computer science and engineering majors must prove many competencies, but can graduate without taking even one course in the design of a modern, relational database (RDB). Then, their first job willrequire them to interface with a large RDB of some kind; ALL important facts are kept in them.

Using a spreadsheet like Excel can be learned by the average computer user. Not so with a relational DBMS like Oracle, or even Access. One first must learn the relational model, and data modeling, then DB design and somethings about DB administration. This takes at least one course.

FURTHER DESCRIPTION OF WS#5

This workshop will assemble principles from experts in software engineering who will attest to this thesis: the biggest step in writing smart & sharable code is imagining the data structures and their transformations via the algorithms that are written to implement the program's purpose, to solve a problem in symbol manipulation, for example. This was said by the great computer scientist **Charles Simonyi** (BS, UC-Berkeley; PhD Stanford) in the 1986 book, Programmers at Work, MS Press:

Question: "Is that [data structures] the biggest step?

SIMONYI: "Absolutely that is the biggest step... For the most part the code writes itself, but it is the data structures I maintain that are the key."

Every CS major has to learn to use Data Structures; the smarter software engineers, in my 35 years' experience, take DB I to know DBMS, which in effect are re-usable data structures. The workshop will be interactive and involve team exercises in programming problems that were inspired by Jon Bentley's classic book, Programming Pearls.

PROBLEMS ARE INDEPENDENT OF CODING LANGUAGES, OR CODE

ANY smart modern relational DBMS already includes shared code to access the data, as it is kept in standard relational data tables.

So if you design and build a good RDB, using a Relational DBMS, the code is already written!

The problem is reduced to designing the right tables that allow one to solve the problem.

The meta-problem of acceptance

ask your department's curriculum leader(s):

Q: Should we not vary the ACM/IEEE curriculum for our major(s) to reflect the 21st century's big changes in how people compute?

The last 15 years have brought us

mobile devices, tablet computers, search engines across VLDBs, universal WWW

WHY CS HAS TO CHANGE with the Times

Mathematics, the queen of all sciences, and thus of engineering (applied science) needs never to change. It is about eternal truths and proncip

But we do: computers change to better serve mankind, and we exist to educate people how best to use computational devices, that is the most efficient ways to solve their problems.

Non-reusable, non-sharable coding's defunct; that includes 'elegant' O-O programs too

A PROBLEM from PoetRobotics Inc.

* Design a software system - algorithms and data structures - to find All Rhymes of a word.

GIVEN: a partly filled lookup list of basic words often used in poems/songs; for example

WORD	RHYMES
moon	June, boon, croon, tune, noon, spoon, ... =====>

Do we need a row for every word? No; repetitive data & searches. But, fast lookup

WORKSHOP EXERCISE ONE:

DESIGN the data srurctures and algorithms needed to solve the ALL Rhymes problem using an e-dictionary to look for the rhymes of words.

HINT 1: you may need another table that has all phonemes that end all common words e.g. moon ends in -oon; June ends in -une, and in sound, -oon = -une.

HINT 2: Equal phonemes produce same

EXERCISE DISCUSSION #1

OK, TIME TO STOP AND COMPARE NOTES;
APPOINT A SPOKESMAN FOR YOUR GROUP TO COME UP AND EXPLAIN YOUR TEAM'S SOLUTION, FOR FIVE MINS. MAX.

HINT 1 WAS PURPOSELY NOT COMPLETE: OTHER RHYMES WITH moon END IN -EWN [hewn] OR -UGN [impugn] but, if words with these endings are in your side DATA source or basic dictionary, you'll find them. No googling!

Bentley's solution to Workshop exercise one

I believe my limited solution follows the spirit of Bentley's hints at a solution, but in my 1/E of his 1986 book, I saw no full solution, and I have not seen the 2/E of 2000 for this specific problem.

A first question you guys should have asked each other is how close to "All Ryhmes" do we need to be? You can't prove your solution set will find EVERY Rhyme in English's 600,000 words!

ALL-RHYMES ALGORITHM/DATA

0. BUILD A DATA TABLE OF FINAL-SYLLABLE PHONEMEs & WORDs as you go;

1. ISOLATE F-S PHONEME OF INPUT WORD; LOOK IT UP IN SIDE DATA TABLE:

moon => -oon; if no entry, create one for -oon put all the WORDs in its rows into RHYME table

2. EVERY ROW IT IS ON HAS A RHYMING WORD: (-oon, soon); (-oon, boon) etc. PUT these Words into Rhymes, keyed by

HAVE WE RE-INVENTED A WHEEL?

IF YOU GOOGLE 'RHYME FINDER' YOU WILL DISCOVER MANY OFFERS OF SUCH S/W. I TRIED A FREE ONE AND IT WAS NOT PERFECT: FOR OUR SAMPLE WORD IT GAVE "***SHOGUN***" AND WHEN I CLICKED ON THAT WORD IT WENT WAY OFF COURSE = WE NO LONGER HAD -oon RHYMES AT ALL!

[Tch, tch, WriteExpress!]

I GUESS IT USED A SOUNDEX OR SPELL-ALIKE TRICK THAT GOT CONFUSED!

WORKSHOP EXERCISE TWO : PROPERTY TAX DB PROBLEM

PROBLEM: You are given two data tables

ADDRESSES [NAME, ID#, ADDRESS]

PARCELS [ADDRESS, EVAL, 2011_TAX, 2012_TAX];

your problem is to be able to print a combined list of

[NAME, ADDRESS(es), EVALs, LAST_2_TAXES].

INFO: The 2011 levy was 30 mills (.030 x EVAL) and the 2012 levy will be 35 mills (.035 x EVAL)

HINTS: In the ADDRESSES table, ID# is the key but NAME is not (Paul Allen has many properties in Seattle.)

In PARCELS, assume ADDRESS is the unique key.

FIRST DRAW ALL THE NEEDED DATA STRUCTURES!

SECONDARY HINT FOR E-R DESIGN

1. DRAW THE E-R MODEL FOR THIS DATA

1. USE THE ERD TO MAKE A RDB THAT HAS A THIRD TABLE TO HOLD ANSWERS

[If you calculate the 2012_TAX from the 2011_TAX, assuming the EVAL is constant, it is slightly faster arithmetic: 2012s = 1.2 x 2011s]